12-08-2019

Docente: Juan Pedro Cardona Salas
Materia: Lenguajes de Computación III

Java

Clases
system
Gráficos
Bases de datos
Archivos

SQL

Altas, bajas, lectura y cambios

Parciales

33% { Proyecto 40%
Examen-proyecto 40% → Teoría
Participación 20%

33%

34%

13-08-2019

Java es un lenguaje de propósito general orientado a objetos, pero mantiene datos primitivos (desventaja porque hay que hacer conversiones)
¿Cómo lo hace Java? Anteriormente, C o Fortrand, todo el programa lo compila en un ejecutable .exe
Java lo que hizo fue trabajar un archivo x.java que trabajo con JVM (Java Virtual Machine), esta es una máquina virtual que se encarga de cargar clases y una especie de compilación

Otro elemento es JRE (Java Runtime Environment) es un entorno que interpreta lo que el JVM te manda

x.java $\xrightarrow{JVM}$ x.class $\xrightarrow{JRE}$ x.exe

Todo esto es independiente del sistema operativo
Los anteriores manejan memoria, optimiza el código (Garbage Collector inutiliza las variables sin uso) pero la primera compilación es más lenta

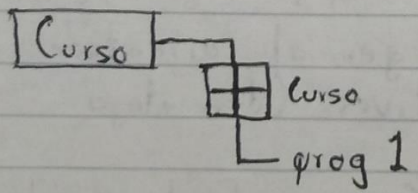
Otra ventajo es que tiene una familia de clases estándar

Instalación de Java

Java + Netbeans 8.0 +

| | | |
|---|---|---|
| SE | ✓ | Standard Edition |
| EE | ✗ | Web Enterprise Edition |
| ME | ✗ | Mobile Edition |

En Netbeans:
File → New Project → Java Application




Con 3 variables numéricas

```
package curso;
public class prog1{
    main{
        int a=8, b=6, c=7
        if(a==b && a==c)
          {
```

```
        system.out.print("El triangulo es equilatero");
        }
      else
        {
        if(a==b||a==c||b==c)
          {
          system.out.print("El triangulo es isosceles");
          }
        else
          {
          system.out.print("El triangulo es escaleno");
          }
```

Crear un vector de 10 elementos e imprimir el mayor de ellos

```
int[] arr;      // Declaración
arr = new int[10]; // Instanciación
arr[0]=4; arr[1]=9; arr[2]=3 ... arr[9]=7;
 int mayor = -100;
for(int=0; i<10; i++)
{
    if(mayor<arr[i])
    {
      mayor=arr[i];
    }
}
system.out.print("El mayor es " + mayor);
```

Si ponemos sop(arr.length) ⟶ 10

3 4 5 Prim
6 7 8

14-08-2019

JVM hace
- Carga clases
- una especie de compilación
- Optimiza código (no entra virus)
- Administra memoria

Java tiene tipos primitivos y objetos
Un objeto puede cambiarse a string, a Integer y longitud
Los primitivos pueden cambiarse a otros primitivos y a eso se le llama typing o tipeado

El Garbage Collector tiene mejores usos con los objetos "aquellos declarados como new" pero los primitivos son permanentes

Haga un programa en Java que lea un número entre 1 y 12 e imprima la estación

```
import java.io.InputStreamReader
public class mes
{
    public static void main()
    {
        char c=' ';
        BufferedReader br= new BufferedReader (new
InputStreamReader (System.in));
        Sop ("Dame mes");
        try{c=(char) br.read(); // cast
        } catch (IOException e) {Sop ("ups"); }
```

```
String season = "nada";
switch (c)
{
    case 12:
    case 1:
    case 2:  season = "Invierno"; break;
    case 3:
    case 4:
    case 5:  season = "Primavera"; break;
    case 6:
    case 7:
    case 8:  season = "Verano"; break;
    case 9:
    case 10:
    case 11: season = "Otoño"; break;
}
Sop("La estación es " + season);
```

## Objetos

Es una estructura que nos permite manejar variables y métodos en una sola variable, es una manera de agrupar y administrar las partes de un programa, su antecedente es el struct del C

Las variables ahora son atributos
Los métodos ahora son conductas

Haga un programa que cree la clase nube y determine cual llueve más, definida por tamaño y densidad

java class

```
class Nube
{
    int tamaño;
    int densidad;

    int lluvia ()
    {
      return tamaño * densidad;
    }
    :
```

Constructor: Es un método que nos permite asignar valores a los atributos en el momento de crear el objeto

```
    :
    Nube (int tam, int den)
    {
      tamaño = tam;
      densidad = den;
    }
```

java main class

```
class test Nube
{
    main()
    {
      Nube n1 = new Nube (15, 21);
      Nube n2 = new Nube (17, 19);
```

```
        if (n1.lluvia == n2.lluvia)
            Sop ("Iguales");
        if(n1.lluvia > n2.lluvia)
            Sop ("Nube 1 es mayor");
        if(n1.lluvia < n2.lluvia)
            Sop ("Nube 2 es mayor");
    }
```

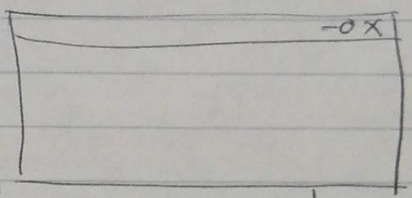
Puede leerse un dato con el br.readLine();
donde br es un objeto de tipo BufferedReader

O con el Scanner s = new Scanner (System.in);
(El clásico de Efra)

## Objetos

```
Clase   objeto = new Clase ();
Integer i = new Integer(10);
```

Desechable `new objeto();`




Esta forma de crear objetos es cuando no necesitamos referenciarla con un nombre de variable, el ejemplo es una ventana (JFrame o JPanel)

## Conversiones

```
int a;
double b = a;        } Widening

double a;
int b = (int) a;     } Narrowing
```

```
Integer ob = new Integer("30");   } Unboxing
int i = ob.intValue();

int i = 10;
Integer b = new Integer(i);       } Autoboxing
```

Convertir un char primitivo a un int primitivo y un String

```
char c = '5';
int i = Character.getNumericValue(c);
String s = Character.toString(c);
```

Convertir de String a int e Integer

```
String ss = "44";
int i = Integer.parseInt(ss);
Integer y = Integer.valueOf(ss);
```

De primitivo a string

```
int b = 123;
String s = Integer.toString(b);
```

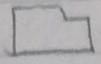
java.awt tiene todo el paquete gráfico
javax.swing es el nuevo paquete pero usa el anterior

En el código en 7, extends es "hereda de"
implements es "usa" y ActionListener es un manejo de eventos
En 22 usamos una etiqueta para optimizar en 23
En 27 se hacen conversiones
En 38 creamos el objeto de la clase

```
1: package javaapplication5; // Carpeta de la aplicación
2:
3: import javax.swing.*;
4: import java.awt.*;
5: import java.awt.event.*;
6:
7: public class f_JPanel_01 extends JPanel implements ActionListener {
8:
9:     JButton  confirma1;
10:    JTextField texto1, texto2;
11:
12:    public f_JPanel_01() {
13:        texto1 = new JTextField(5);
14:        texto2 = new JTextField(15);
15:        confirma1 = new JButton("  cuadrado ");
16:        confirma1.addActionListener(this); // Método va al 23
17:        setLayout(new FlowLayout());
18:        add(texto1);
19:                add(texto2);
20:                add(confirma1);       }
21:
22: @Override
23: public void actionPerformed(ActionEvent ae) {
24:    if (ae.getSource() == confirma1) {
25:        Integer i=new Integer(texto1.getText().toString());
26:        int cua =  i.intValue()*i.intValue();
27:        Integer ii = new Integer(cua);
28:                texto2.setText("respuesta "+ii.toString());
29:                                  }       }
30:
31:      public static void main(String[] args) {
32:
33:        JFrame.setDefaultLookAndFeelDecorated(true); // Apariencia
34:
35:        JFrame frame = new JFrame(" dos textfield y un boton ");
36:        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37:
38:        f_JPanel_01 bot = new f_JPanel_01();
39:        frame.getContentPane().add(  bot  ,BorderLayout.CENTER);
40:
41:        frame.setBounds(100,100,400, 325); // Dimensiones
42:        frame.setVisible(true);
43:    }
44: }
```

26-08-2019

Semana 3 de 16

package 

import swing
awt
awt.event
} Elementos visuales

extends hereda
implements usar las clases de eventos

confirma 1% addActionListener ⟹ Hace el evento

actionPerformed(ActionEvent ae) ⟹ Ejecuta metodo event

- Hacer un programa que capitalice

```
actionPerformed( )
{
    String a = text1.getText(); toString();
    text2.setText(a.subString(0,1).toUpperCase()
+ a.subString(1));
}
```

Hacer un programa que diga si una palabra es palindromo o no

```
String a = text1.getText().toString();
StringBuilder b = new StringBuilder(a);
if(a.equals(b.reverse().toString()))
    Texto2.setText("Es palindromo");
```

a

```
String Tokenicer s1 = new String Tokenizer ("String", " ")
while (s1.has More Tokens ())
{
    String t = s1.next Tokens();
    Sop (t);
}
```

```
void Action Performed(...)
{
    if(...)
    {
        BString Tokenizer st = new String Tokenicer (texto1.getText(), " ");
        while (st.has More Tokens ())
        {
            String s1 = st.next Token();
            Character s2 = s1.char A (0);
            Texto2.setText(s2.to String().to Upper Case...
```

Programa que sume

```
void Action Performed(...)
{
    int suma = 0;
    String Tokenicer st2 = new String Tokenizer (text1.getText(), ",")
    while (st2.has More Tokens())
    {
        String.s1 = st2.next Token();
        Integer i = new Integer(s1);
        suma = suma + i;
    }
    text2.setText(suma.to String());
```

Programa que separe pares y nones

Semana 4 de 16 02-09-2019

Metodos Estáticos (variables o clases estaticas)
Propiedad que no requiere instanciación

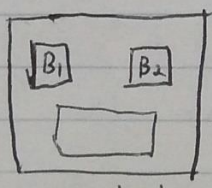
Integer i = new Integer();
instanciación

Por ejemplo:
a=5 b=3
Sop("Math.max(a,b)); ⇒ Math no se instancio

Herramienta de Interfaz Gráfica
Netbeans
new -> File -> JFrame
JPanel

initComponent no es parte de Java, es de Netbeans
gaps sirve para mantener la misma distribución de los componentes aunque cambiemos el tamaño

Tarea: Hacer




A1 clickear B1 : B1
A1 clickear B2 : B1, B2
A1 clickear B1 : B1, B2, B1

Examen: Gato con botones

09-09-2019

Colecciones
Java provee un conjunto de colecciones, también conocidas como estructuras de datos
Las colecciones pueden usarse como objetos comunes, pero el Standard de programación es utilizarlo con interfaz, así se cumple el principio de programación y Programming for interface

que ayuda al mantenimiento y mejora la flexibilidad

| Interface | Implementations = Clase | |
|---|---|---|
| List | Array list | Lista Dinamica con indice |
| | Linked list | Lista ligada |
| | Vector | [i] |
| | Stack | [i] LIFO |
| Set | TreeSet, HashSet, LinkedHashSet | pares clave/dato |
| Map | TreeMap, HashMap, LinkedHashMap | |

Hacer programa

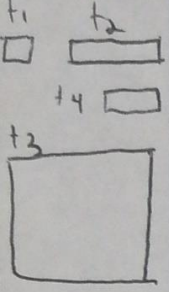
Componentes

```
JTextField t1, t2; int i=0;
List list = new ArrayList;

action 1
{
   list.add(t1.getText());
   t1.setText("");
   t3.setText(null);
   i=0;
   for(String str : list)
   {
      t3.append(i + " " + str + "\n");
   }
}

action 2
{
   i= Integer.parseInt(t2.getText());
   list.remove(i-1);
}
```

Nota: Las colecciones no almacenan datos primitivos, sólo objetos y van entre simbolos < >




Antes era
clase a = new clase();
Ahora es:
clase a = new interface();

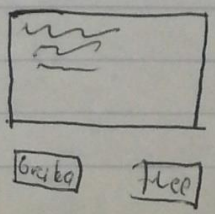
Nota Haga un altas y bajas de nombres que siempre mantenga su número. La colección que permite número y dato es HashMap() y la interface Map

Map<Integer, String> dir = new HashMap<>();

Action Performed t2
{ Integer i = new Integer(t1.getText());
dir.put(i, t2.getText());
Iterator iter = dir.entrySet().iterator();
while (~~has Next~~ iter.hasNext()) {
Map.Entry entry = (Map.Entry) iter.next();
t3.append(entry.getKey().toString() + " " + entry.getValue()
+ "\n"); }}

Action Performed t4
{ dir.remove(Integer.parseInt(t4.getText()));
// de Iterator al final de ActionP t2 es igual

Ejercicio Haga un programa con textArea, que grabe y lea



try
{

```
Action Performed Graba
{ BufferedWriter bf = new BufferedWriter(new FileWriter(
"c:\\archivo.txt"));
    String a = textArea1.getText();
    String[] arrS = a.split("\n");
    for(int b=0; b<arrS.length; b++)
    {     bf.write(arrS[b]);
          bf.newLine(); }
    bf.flush();
    bf.close();  } catch(IOException e)
                 { e.printStackTrace(); }}

Action Performed Lee
{ textArea1.setText(null);
  try { FileReader fr = new FileReader("c:\\archivo.txt");
        BufferedReader br = new BufferedReader(fr);
        String line;
        while((line=br.readline()) != null
        { textArea1.append(line + "\n"); }
        br.close(); fr.close();
      } catch(IOException e) ... igual }}
```

Colecciones

con Interface  List<Integer> lis = new ArrayList<Integer>

sin Interface  ArrayList<Integer> lis = new ArrayList<Integer>

Interface: Práctica estándar de programación donde es más fácil modificar o sobrecargar métodos en la interface que en la clase original

Agrega elementos

| Interface | Genérica |
|---|---|
| List | ArrayList, LinkedList, Stack, Vector |
| Set | HashSet, LinkedHashSet, EnumSet |
| SortedSort | TreeSet |
| Queue | PriorityQueue |
| Deque | LinkedList, ArrayDeque |
| Map | HashMap, Hashtable, LinkedHashMap, EnumMap |
| SortedMap | TreeMap |
| Otras interfaces: | List, Set, Map, Stack, Queue, Deque, Deque, Iterator, Iterable |

```
LinkedList<Integer> n = new LinkedList<Integer>;
        Integer i = n.getFirst();
                  .getLast();

list String.set (2, "hola");


ActionPerformed{
    String a1[] = JTextField1.getText().split(",");
          a2
    HashSet<String> set = new HashSet<>();
    set.addAll(Array.asList(a1));
                            a2

    JLabel1.setText("Unión: " + set.toString()); }
Action P {
    String a1...
    String a2...
    Set<String> inter = new HashSet<>(Array.asList(a1));
    inter.retainAll(Array.asList(a2));
    JLabel2...      "Interseccion: " + inter.toString()); }
Action P {
    String a1...
    String a2...
    Set...          diff
    diff.removeAll
    JLabel3...      "Diferencia: " + diff.toString()); }
```

Metodos de Iteracion en Collections

```
for(int i=0; i<list.size(); i++){
    String aName.get(i);
    Sop (aName)}
```

```
Iterator<String> it list.iterator();
while(it.hasNext()){
String aName = it.next();
Sop(aName); }
```

Como el String Tokenicer

```
Map<Integer, String> map1 = new HashMap<>();
Iterator<Integer> = map1.keySet().iterator();
while(it.hasNext()){
    Integer key = it.next();
    String value = map1.get(key);
    Sop(value); }
```

Metodos estaticos de Colecciones
No necesitan instanciar

- Math.max(a,b);
- Integer.parseInt(a);
- Collections.sort(list);
- Arrays.sort(a);
- Collections.copy(lista, listb));
- Collections.reverse(lista);
- Collections.shuffle(lista);
- List<String> sub = list.sub(),5);
  sub.add("5");

Ultimos Comentarios sobre Collections

List más rápido que ArrayList

Set contiene elementos no repetidos { HashSet / TreeSet ✓

Map es el más usado y versátil. Se puede usar para ver la correspondencia entre un nombre y un número

SQL Structured Query Language

Lenguaje de búsquedas para bases de datos relacionadas

- MySQL
- Access
- Ingress
- Derby

Es una base de datos de comandos

La base es un entero

Tablas como archivos

```
Select id, nombre from nomina
Select * from nomina
Select * from nomina where edad >20
Select * from nomina where nombre like "Ma%"
Select count(*) from n
Select avg(edad) from n
Select max(salario) from n
Select nombre from n order by nombre desc/asc
Select * from n where nombre in ("juan", "luis")
Select * from n where edad between 10 and 20
Insert into n (id, nombre, edad, salario) values
 (8, Arturo, 40, 10,000);
update n set edad = 20 where edad = 69
delete from n where id = 4
```

Hacer programa que lea contraseña con 5 oportunidades

```
pass = JPasswordField(12);
pass.setEchoChar('*');
   :
ActionPerformed{
 if(pass.getPassword().equals("pass")
      Sop("Nice");
 else{ op--;}
 if(op==0){ System.exit(0); setVisible(false);
```

04-10-2019

Datos

Altas Bajas Cambios — .txt

MySQL — Base de Datos

Interface
- ODBC
- JDBC

ODBC: Panel Control > Herramientas Admin > Tipo | Data

JDBC: NetBeans > Libraries > MySQL

MySQL

▲ SQL start stop → Country Sample →

Instalar MySQL

```
crear  database  test
       tabla     test1
java   insertar
java   leer
```

---

```
import java.sql.connection   // JDBC
       .DriverManager // administra la conexión
       .ResultSet // collections que almacena los querys (selects)
       .Statement // genera y ejecuta instrucciones SQL

public class lee{
  main{ try{ Class.forName("com.mysql.jdbc.Driver");
      // ejecutar archivo de librería MySQL Connector libraries add
    Connection con = DriverManager.getConnection("jdbc:mysql//
                                                  driver
```

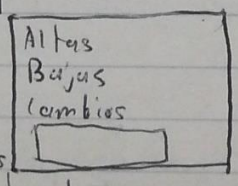
```
localhost:3306/test", "root", "0987");
   puerto   base    propietario  password

Statement stmt = con.createStatement();
   // crea statement (instruccion) asociado a conexion (con)
ResultSet rs = stmt.executeQuery("select * from test1");
                                                  mi tabla
while(rs.next())
   Syso(rs.getInt(1) + " " + rs.getString(2) + " " + rs.get
                id                  nom
Int(3));
    edad
   con.close(); } catch(Exception e){
      Syso(e); }    }}
```

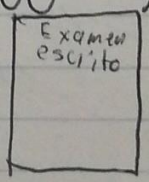
Los 4 import iguales

```
public class inserta { main {
   int id=20; String nom="juan"; int edad=20;
   try{ Class.for...
      Connection con...
      String inser = "insert into test1 values(" + id +
",'" + nom + "'," + edad + ");";
      stmt.executeUpdate(inser);
      con.close(); } catch ...
```

Proyecto
Lunes lab 61
Collections
↓
SQL Base de datos
ID Nombre



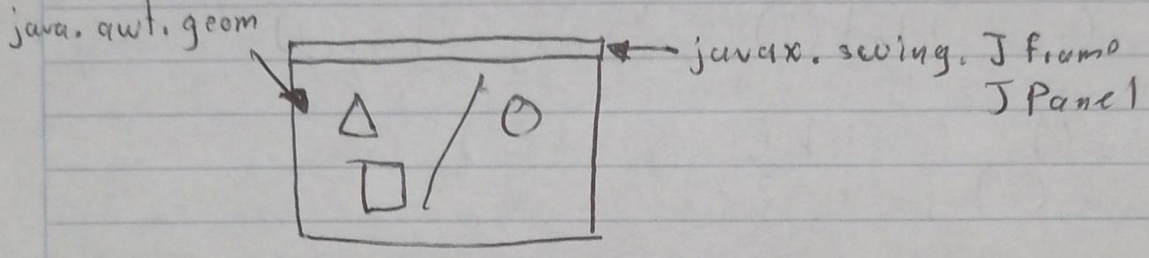

Examen
Miercoles lab 61
Aula Virtual exam2.txt
id, num, edad, salario, altura → Sql
15,000 20,000 datos



Drawing

AWT libraries are set of classes provide to draw shapes on a window

java.awt.geom AWT Abstract Windowing Toolkit




Modelos

```
public void paint (Graphics g){
    super.paint(g);
    g.drawOval(150, 150, 50, 50);
    g.drawLine(10, 10, 150, 150); }
```

(x, y, width, length)

Otra forma

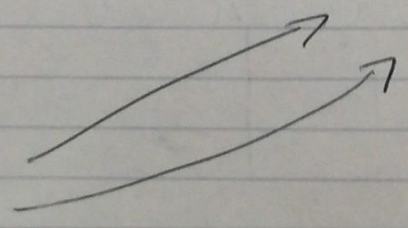
```
paintComponent (Graphics g){
    Graphics2D ...
```

06-11-2019

Gráficos JFreeChart → De otra persona → third Party

Usando JFreeChart

Library add JAR/Folder




import 8

import 2

```
public pieChart
    creq DemoPanel();    // JPanel
```

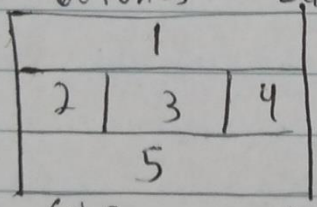
dataSet ((ollection) // var, txt, sql
Create Chart (tipo_de_grafico, dataSet, true, true, false);

instancia de DemoPanel()
main
setVisible(true), setSize();

tutorials Point ; Free Chart
8 ejemplos → tamaño 2

.txt y bases de datos

Mañana Examen 2 puntos
a) 5 botones Layout BorderLayout



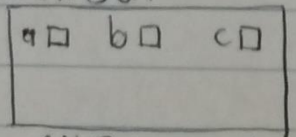

Syso("Presionaste 1");

b) GridLayout (3 3)

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Syso ("Presionaste 6");

c) CheckBox




Syso ("Presionaste c");

Problema: Código feo → es difícil de usar o reusar

Error 1

En Gráficos:

Todo en main

Sin JFrame o ActionListener

Solución Separar

Tarea: 1: Programa de checkbox pasarlo de "Código feo" a ~~solución~~ MVC (modelo, vista, controlador)

2: Hacer el programa de choice

Error 2

En código:

Para cerrar son 3 líneas en Frame y 1 en JFrame

(Thread = hilo de ejecución para programar con varios procesos a la vez)